SHAPEIOT: SECURE HANDSHAKE PROTOCOL FOR AUTONOMOUS IoT DEVICE DISCOVERY AND BLACKLISTING USING PHYSICAL UNCLONABLE FUNCTIONS AND MACHINE LEARNING

Cem Ata Baykara[1], Ilgın Şafak[2] and Kübra Kalkan[1]

[1]Department of Computer Science, Ozyegin University, Istanbul, Turkey
[2]Fibabanka R&D Center, Istanbul, Turkey

ABSTRACT

This paper proposes a new lightweight handshake protocol implemented on top of the Constrained Application Protocol (CoAP) that can be used in device discovery and ensuring the IoT network security by autonomously managing devices of any computational complexity using whitelisting and blacklisting. A Physical Unclonable Function (PUF) is utilized for the session key generation in the proposed handshake protocol. The CoAP server performs real-time device discovery using the proposed handshake protocol, and anomaly detection using machine-learning algorithms to ensure the security of the IoT network. To the best of our knowledge, the presented PUF-based handshake protocol is the first to performs blacklisting and whitelisting. Whitelisted IoT devices not displaying anomalous behavior can join and remain in the IoT network. IoT devices that display anomalous behavior are autonomously blacklisted by the CoAP server and are either disallowed from joining the IoT network or are removed from the IoT network. Simulation results show that amongst the five machine learning algorithms studied, the stacking classifier displays the highest overall anomaly detection accuracy of 99.98%. Based on the results of the network simulation performed, the CoAP server is capable of blacklisting malicious IoT devices within the network with perfect accuracy.

KEYWORDS

IoT Networks, Network Security, Handshake Protocols, Anomaly Detection, Machine Learning

1. INTRODUCTION

IoT has gained immense mind share in both academic and industry alike over the past several years. In our everyday lives, IoT enables devices to be aware of their surroundings, efficiently communicate, and ultimately create a better environment for the people [1]. Devices from the same owner effectively forms a smart environment for that owner where each device can communicate and effectively combine their strengths to overcome their weaknesses [2]. To be able to recognize and utilize the potential of IoT, secure discovery and access control is essential [2]. However, these devices have differing computational complexities, and not all of them are equipped with means to prevent themselves from malicious malware. Many IoT devices are manufactured with inherent security vulnerabilities [3]. These vulnerabilities often pose a risk for the security of an entire IoT network that includes vulnerable devices[4]. In a centralized IoT network, it is often the responsibility of the central entity to ensure the availability and security of the network[5].

This paper proposes a lightweight and secure handshake protocol that is computationally inexpensive and suitable for detection of IoT devices of any computational complexity for the purpose of device discovery and device management in an IoT network. A Physical Unclonable Function (PUF) is used in securely generating a session key by the IoT device. An autonomous anomaly detection, whitelisting and blacklisting approach is proposed to prevent unwanted or malicious devices from joining, re-joining, or remaining in the IoT network. IoT network simulation results are provided of the proposed handshake protocol, where the CoAP server in the network performs real-time autonomous anomaly detection using pre-trained machine learning (ML) algorithms. To the best of our knowledge, the handshake protocol proposed in this paper is the first to address both the computational complexity and security challenges of IoT devices in the device discovery utilizing PUFs and a whitelisting and blacklisting approach in autonomously ensuring IoT network security. The contributions of this paper are as follows.

1. A new lightweight handshake protocol implemented on top of CoAP that utilizes PUFs to generate the secure session key, in addition to a whitelisting and blacklisting approach in ensuring IoT network security is proposed. To the best of our knowledge, this is the first PUF-based handshake protocol that performs blacklisting and whitelisting. The computationally inexpensive property of PUFs allows even the most basic IoT devices to access an IoT network by using the handshake protocol.
2. The proposed protocol allows central or distributed authorities of the IoT network to establish negotiated communications, provide novel services, and autonomously prevent undesired devices from joining, re-joining, or remaining in the IoT network using whitelisting and blacklisting.
3. This paper proves that the proposed handshake protocol is secure against our threat model, including Man in the Middle (MITM), forgery and replay based attacks, with a security analysis.
4. The paper presents an IoT network simulation to test the effectiveness of the proposed handshake protocol. The simulation includes a CoAP server that performs real-time anomaly detection using pre-trained machine learning algorithms. The paper also provides different machine learning classifiers that can be used and presents a detailed discussion and analysis of which classifier ispreferable.
5. Based on the analysis and network simulation performed, the handshake protocol proposed in this paper can easily be adapted to real-life IoT networks.

The remainder of this work is structured as follows. Section 2 presents the existing approaches and related work. Section 3 presents a description and the security analysis of the proposed handshake protocol in detail. Section 4 presents the experimental work performed, namely the network simulation and its results. Finally, Section 5 discusses the conclusions of this work.

2. RELATED WORK

This section provides a summary of related work and discusses the differences between the proposed handshake protocol.

In the Transport Layer Security (TLS) protocol, session keys are randomly generated by the client [6]. This approach is prone to replay and forgery attacks, which is dependent on how the client generates the random string. Compared to the TLS protocol, the proposed handshake protocol in this work offers a more lightweight and secure approach to mutual authentication by generating session keys using a PUF and utilizing a whitelisting and blacklisting approach in autonomously ensuring IoT network security using machine learning.

An approach to using PUFs to address the problems of low computational power of IoT devices is proposed in [7]. This work shows in detail the benefit of using PUFs to establish a secure session with IoT devices by comparing their proposed protocol with other existing solutions such as Datagram Transport Layer Security (DTLS) handshake protocol and User Datagram Protocol (UDP). The results indicate that with the use of PUFs, the authentication process results in a reduction in power of up to 45% by also using 12% less memory, compared with the existing solutions listed before. The authors propose a method of detecting artificially generated challenge-response pairs (CRPs) by using neural networks that their proposed verifier (server) uses to authenticate the nodes. By employing this technique, the server can detect malicious nodes that try to authenticate with the server by using replayed CRPs. However, the protocol can only authenticate existing and trusted nodes in the network; it does not utilize whitelisting or machine learning techniques for autonomous blacklisting of suspicious devices as proposed in this paper. Thus, it is not suitable for general and public use since it constrains the system from allowing new nodes to join the network. In contrast, the protocol presented in this paper, while keeping a record of previously explored nodes, allows new nodes to join the network and uses whitelisting and blacklisting techniques in ensuring the security of the IoT network.

Lightweight key exchange protocols that use pre-shared secret symmetric keys are proposed in [8]-[9]. These protocols assume that one or more symmetric keys are readily available to the client and the server before the handshake protocol begins. However, this is not a secure or scalable approach since it requires every new client to obtain the shared keys a forehand and is prone to forgery and brute force attacks. In comparison, the proposed handshake protocol generates a secure session key on-the-fly without requiring pre-shared symmetric keys, as well as utilizing whitelisting and blacklisting in ensuring the network security.

Lightweight key exchange protocols using mutual authentication are proposed in [10]-[11]. These protocols use PUFs not only for generating session keys, but also for registration and authentication purposes. However, these protocols do not study autonomous blacklisting techniques in ensuring the network security. The protocols store long-term keys to authenticate the server or a device, a process which requires extensive computation and data storage. Given the limited computational capability of IoT devices, the proposed protocol in this paper tries to minimize the computation required from the IoT nodes for server authentication. Thus, the proposed protocol in this paper authenticates the server with the help of a Certificate Authority (CA). However, using CA has its downsides as it requires the assumption that the CA will always be available since the protocol cannot be completed otherwise. Using CA for authentication also increases the communication flow required to perform the handshake. The proposed protocol in this paper performs the key exchange with 7 communication steps, whereas the protocol proposed in [11] only needs 3 communication steps for a registered user. However, unlike these PUF-based authentication and key exchange protocols, the protocol presented in this paper lets any user to initiate the key exchange without client registration or client verification processes.

Similar handshake protocols are proposed in [12]. These protocols also utilize the low computational complexity of PUFs for IoT devices and provide detailed analysis in terms of computation, memory, and communication overhead. The results in [12] show that using PUFs for secure session generation is a suitable solution for IoT devices. Another protocol proposed in [13] can generate a secure session between IoT devices. This protocol uses a trusted and pre-authenticated server for secure session generation, similar to the handshake protocol proposed in this paper, which uses a CA for server authentication. Results in [13] are also in agreement with the previous related work on showing the effectiveness of using PUFs instead of using existing solutions. However, even though authors of [12] state that their proposed protocols can be adapted to real-life use cases, they do not provide a real experimental setup or a real time simulation to prove this statement. In contrast, this paper provides the results and details of a real time network simulation using the handshake protocol proposed to show that it can easily be adapted to real life use cases.

[14] proposes a secure PUF based authentication and identity-based key exchange protocol suitable for a distributed IoT network. It differs from this work in that a certificate-less identity based key exchange approach is used, and that it does not study whitelisting and blacklisting in ensuring the network security.

[15] proposes a lightweight mutual authentication protocol based on a new public key encryption scheme that uses the encryption scheme to transmit challenges and check whether the recipient can respond accordingly. The protocol is shown to have a performance significantly better than existing RSA and ECC based protocols. It does not require the use of a CA for authentication, as proposed in this paper. Additionally, whitelisting and blacklisting using ML techniques in ensuring the network security are not studied.

In [16], a new lightweight mutual two-factor authentication mechanism is proposed, where an IoT device and server authenticate each other and establish a key exchange usingPUFs and a hashing algorithm. The main difference between this paper is that a CA is not utilized in mutual authentication, and whitelisting and blacklisting in ensuring the network security is not considered.

[17] and [18] propose variations of the Datagram Transport Layer Security (DTLS) handshake protocols for IoT networks. [17] proposes a simplified version of the Datagram Transport Layer Security (DTLS) handshake protocol suitable for IoT devices for a general scenario of end-to-end communications based on software-defined networking (SDN). A controller is utilized in generating a symmetric key dynamically, then encrypting and distributing the key to two communicating IoT devices. Certificate verification is shifted from the IoT device to the more powerful controller, where the controller replaces the DTLS server to make a cookie exchange with the DTLS client. The computational overhead and the energy consumption in the IoT devices and the overall duration of the handshake protocol are shown to reduce.[18] separates the DTLS protocol into the handshake phase and the encryption phase, which is shown to enhance the performance in both the device and the network by using a way to delegate the DTLS handshake phase. The proposed scheme supports secure end-to-end communication despite using delegation. However, neither paper utilizes whitelisting or blacklisting methods in ensuring the network security, as proposed in this paper.

[19] proposes a different fingerprinting approach for keeping a log of previously known clients and detect malware and other malicious processes trying to initiate a secure connection using the TLS handshake protocol before the secure session is established. The fingerprinting technique proposed in [19] uses the initial unencrypted hello message sent by the clients after the TCP connection is established. Since this message is unencrypted, TLS fingerprinting extracts metadata presented in the message and generates a fingerprint string using a pre-defined schema. After generating a fingerprint, the server then maps it to the client by keeping a dictionary of known fingerprint to client mappings. By doing so, the server can detect previously known or blacklisted clients trying to initiate a connection and take actions accordingly. However, it is stated that the TLS fingerprinting technique is not sufficient per se to profile clients effectively. A single fingerprint may map to tens or hundreds of unique clients. Thus, the TLS fingerprinting itself is often a poor indicator and additional information is required to increase its performance. The handshake protocol presented in this paper uses PUFs for both client fingerprinting and

session key generation. In order to verify the identity of a client, the server may log the challenges and the session keys used to create sessions and authenticate clients by sending the logged challenges. Since the clients in the proposed handshake protocol use their PUFs to generate the session keys, the client is expected to generate the same session key given the same challenge where only a specific client can create that unique session key due to how PUFs work. This approach is both more precise compared to [19], and it does not require additional information about the client for fingerprinting. Hence, it is more lightweight than TLS fingerprinting technique in [19] which puts additional computational overhead on the server.

3. HANDSHAKE PROTOCOL

CoAP is a specialized internet application layer protocol that allows constrained nodes to communicate with the Internet or with each other in a lightweight way suitable for IoT devices [20],[21], [22], [23]. CoAP provides a request, and response-based interaction model between nodes, while supporting built-in discovery of services and resources. It is designed to easily interface with HTTP to be used on the Web. The key features of CoAP that encouraged us to implement the proposed handshake protocol on top of it includes, but are not limited to [23]:

1. Web protocol fulfilling machine-to-machine (M2M) requirements in constrained environments.
2. Asynchronous message exchanges.
3. Low header overhead and parsing complexity.

PUF is a physical object that for a given input (challenge) produces an unclonable output (response) that serves as a unique identifier for that specific object [24]. The response generated by the PUF can also be called as the digital fingerprint of that object. This concept is often achieved by a semiconductor device such as a microprocessor.

The inimitable feature gives PUFs an advantage over other hardware-based security concepts as even if the attacker has physical access to the device, they cannot clone its intrinsic properties [24]. Thus, PUFs enable us to perform device identification and authentication in a secure manner. It is also stated in [24] that PUFs provide a low-cost alternative when compared with the conventional methods for cryptographic key generation, making them a suitable solution for low complexity IoT devices. The handshake protocol presented in this paper utilizes PUFs to generate the secure session key.

3.1. System and Adversary Model

3.1.1. System Model

The ideal model proposed in this paper assumes that the protocol is used to serve all devices ranging from high complexity power to lightweight IoT devices. The proposed ideal model scheme is depicted in Figure 1. The IoT devices generate the network traffic with the CoAP server after establishing the secure session using the proposed handshake protocol. The malicious devices are assumed to be able to listen every message within the network traffic and can also connect to the main server like other devices. All devices perform the key exchange protocol to achieve secure communication with the server. After concluding the handshake protocol, devices proceed to use the CoAP for secure communication. The proposed model does not assume initially trusted devices, so any device that can provide the necessary information during the key exchange protocol obtains a secure key to communicate. On the other hand, devices must verify the legitimacy of the server by sending the server's certificate to the CA during the handshake protocol. This verification process is assumed to be secure and encrypted between the device and the CA and its security is not within the scope of this paper.

This delegated approach for IoT devices provides efficiency and consistency, which is crucial for use cases that require both communicational integrity and low computational power such as secure payment from IoT devices. The server adds to the IoT network the devices that successfully complete the handshake protocol. The server also performs autonomous real time anomaly detection on the IoT network traffic. Anomalous network traffic, which may pose a threat to the network, is detected by the server and the devices that produce the anomalous traffic are removed from the IoT network and recorded into the blacklist. Further network traffic from the devices within the blacklist is declined by the server.

The proposed model provides a secure and lightweight solution for IoT device communication on CoAP. The key exchange message sizes are kept within the boundaries of the CoAP protocol. The model uses asymmetric encryption and decryption once to establish a secure communication channel with symmetric keys.

3.1.2. Adversary Model

The attacker model provided in[25]is assumed in this paper. The attacker can listen, replay, and create messages in the network, wherethe goals of the attacker are as follows:

- To generate or obtain the key used in the session.
- To acquire device information.
- To obtain the confidential information shared within the communication.

The aim is to provide a secure connection to any device that successfully performs the key exchange protocol. Therefore, the denial of service and jamming attacks are out of the scope for the key exchange protocol analysis.

3.2. Protocol Description

This section presents the proposed handshake protocol in detail. The protocol allows IoT devices to establish a secure session for communication by utilizing asymmetric encryption and physical unclonable functions. Clients and servers that follow the protocol can securely generate a one-time session key to achieve end-to-end encryption for secure communication. Definition of acronyms used within this section are provided in Table 1.

The server makes a lookup on the blacklist for the C_{ID} received from the client. If the C_{ID} is not blacklisted, the server generates a random nonce *SN* and XOR’s this with the CN_0 to generate ch. The server then sends a response to the client’s initial hello message by a hello message of its own including *SN*, *ch*, and its *sc*. Upon receiving the servers hello message, the client first XORs

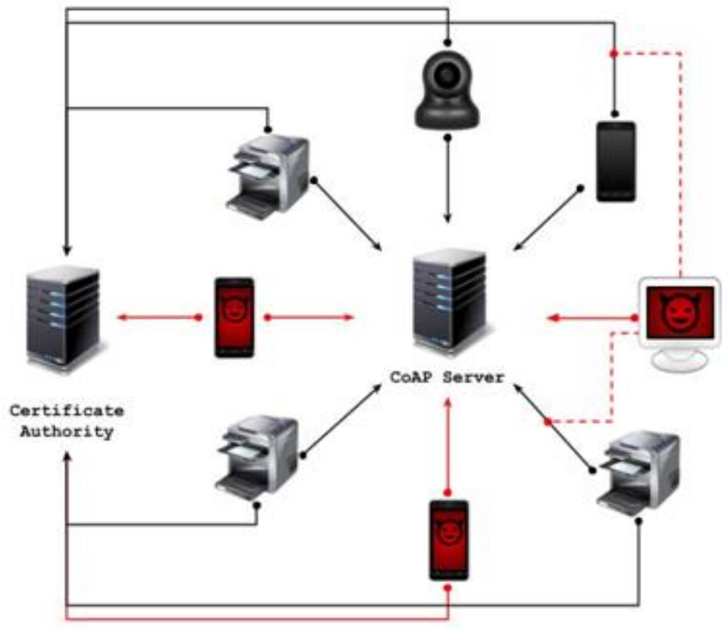

Figure 1. The proposed network model.

the *NS* and the *ch* to reconstruct its CN_0 and check if the server received and used CN_0 when generating *ch*. If the client cannot reproduce its own CN_0 from the ch and NS, the server is not trusted and the client terminates the protocol immediately, otherwise the protocol proceeds as normal. After obtaining the *sc*, the client checks its legitimacy through the CA. The client sends out a message containing the sc to the certificate authority along with NC_1, to show that the communication between the client and the CA is live. The CA responds to the client by sending out a message saying either OK or NOK. If the client receives NOK from the CA, the server is not trusted and the client terminates the protocol immediately, otherwise if the client receives an OK, this means that the server is trusted, and the protocol may proceed to the session key generation phase. The communication between the client and the CA is assumed to be secure (e.g., TLS, pre shared key), and the security between the CA and the client is not within the scope of this paper. After the client confirms the identity of the server through CA, it will now generate the K_S by using the ch. The client uses its PUF to generate the K_S. The client directly uses the ch as an input for the PUF, generating a one-time and non-replicable session key K_S.

Table 1. Definitions and acronyms.

Acronym	**Definition**
CA	Certificate Authority
C_{ID}	Client ID
CN_0	First Client Nonce
CN_1	Second Client Nonce
SN	Server Nonce
ch	Challenge
sc	Server Certificate
S_{ID}	Session ID
K_M	Private Key
K_P	Public Key
K_S	Session Key
N_8	8 Bytes Nonce
D_i	Device Information

The client then extracts the K_P from the *sc* and sends the $K_S$by encrypting it with the K_P of the server. To prevent replay attacks, the client also includes the ch inside this message. The server then decrypts the session key by using its K_M. At this point the server also generates a S_{ID}, by feeding the sum of C_{ID}, *ch* and K_S to the SHA256 function. To ensure that the S_{ID} is unique every time, the server also inserts a random numeric value with a length of 8 bytes to the end of the output given by the SHA256 function. After obtaining the K_S the server sends out a finished message to the client, encrypted with the K_S. This finished message includes the newly generated S_{ID}, to prevent replay attacks. This point is very important as the client will decrypt the finished message of the server to confirm that the server received the K_S, but more importantly that the server decrypted the message 5. If message 6 cannot be decrypted by the client using the K_S, this may signal that the server either cannot decrypt the message 5 and is an untrusted server who does not have access to the K_M, or either the message 5 or 6 has been tampered with. In either of these cases, the protocol is terminated and must be restarted from the beginning. If, however, message 6 can be decrypted by the client using the K_S it sends out a final finished message back to the server which includes its device type and Operating System (OS), and the secure session is generated, and secure symmetric encryption is achieved. The ensuing session is continued using the K_S as the symmetric key.

3.3. Security Analysis

Based on the model described, the security analysis of the proposed protocol is provided in the Section. The likelihood of an attacker breaking the security guarantees of the protocol to achieve his malicious goals is examined, where the analysis is based on the following assumptions:

1. The signature scheme used by the protocol participants (server and client) and the CA is secure (it is impossible for the attacker to forge a signature without the private key).
2. Both the server and client nonce is only picked twice with inconsequential probability.
3. It is not possible to clone or copy the session key generation function used by an arbitrary client.
4. The communication between the client and the CA is secure.

Based on these assumptions, the proposed handshake protocol provides five guarantees.

3.3.1. Guarantee 1

If the protocol is completed successfully, a private session key is generated which is only known by the server and the client. For an attacker to gain access to the session key, the message 5 which is carrying the encrypted session key created by the client must be decrypted. This message is encrypted by the server's public key and can only be decrypted using the server's private key. The only way for the attacker to decrypt this message is by creating the server’s private key. An attacker can never access the private key by assumption 1, so assuming the attacker cannot retrieve the physical storage of the server where the private key is stored, the attacker cannot decrypt the session key.

3.3.2. Guarantee 2

For both the client and the server, the protocol guarantees that mutual authentication is achieved. If an attacker creates a fake server, to proceed with the handshake protocol, he is required to provide the clients trying to connect his server with a signed certificate. Since a client confirms the identity of the server through the certificate authority, the two ways an attacker can pose as the real server is by creating or replaying an OK message in message 4 after the client presents the attacker’s certificate in message 3. The assumption 4 states that the session between client and the certificate authority is secured. The attacker cannot create message 4 for an illegitimate certificate because message 4 is encrypted with unique session keys only known by the real server and the client. If the attacker replays the message 4, an arbitrary client receiving the message 4 will not obtain an OK message after decrypting it with its unique session key, because the encrypted OK/NOK message is unique for every client.

3.3.3. Guarantee 3

The protocol guarantees that a secure and unique session key is generated for each session. The attacker has two options for obtaining the session key: creating the message 5 or replaying the message 5. To replicate the session key for creating the message, the attacker needs the same key generation function and its unique parameters. These parameters are client nonce, server nonce and the timestamp information. Client nonce information is shared in messages 1 and 2. In the case of an attacker capturing both messages and obtaining client and server nonce, the attacker still cannot retrieve any information regarding the timestamp since the timestamp information is never shared in the protocol. By assumption 3, even when the attacker has the client nonce, server nonce and timestamp information, the attacker cannot replicate the key generation process because a PUF is used by the client to generate the session key. For a replay attack to work, the adversary must force both the client and the server to choose nonces to generate the same server challenge. The assumption 2 states that the probability of getting the same nonce values twice is negligible. Hence, the replayed challenge value cannot match with the current session challenge. Therefore, by replaying message 5, the attacker cannot get the expected response in message 6.

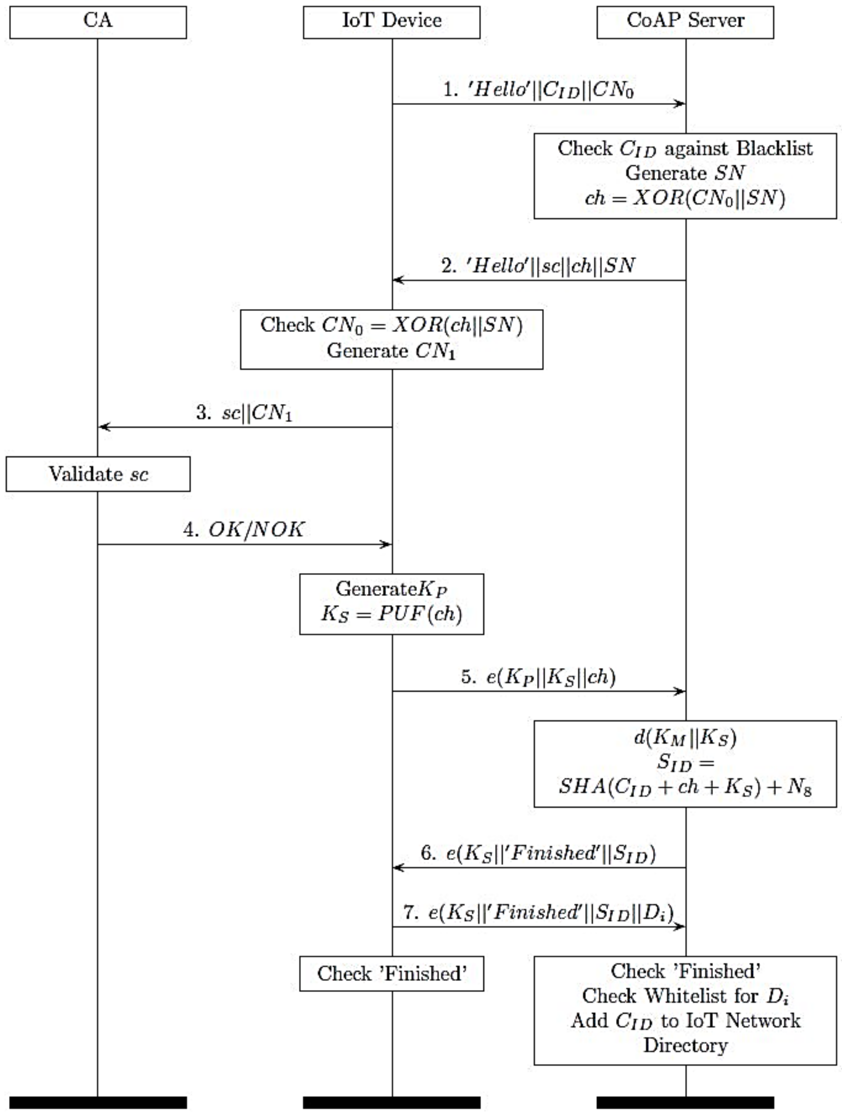

Figure 2. Secure Session Key Generation Protocol using PUF

3.3.4. Guarantee 4

For the client, the protocol guarantees that the generated session is with the trusted server. After sending the encrypted session key to the server in message 5, client now listens for a message that is encrypted with the session key. If this message results in the string “finished” after decryption, client confirms that the server has the session key. An attacker trying to act as the server has two options: replaying the message or creating the message and sending it to the client. Each session has a unique session key that is sent encrypted through message 6. If the attacker replays the captured message 6, an arbitrary client will not obtain the “finished” message after decryption for the similar reasons explained in proof 3. Therefore, the Attacker cannot complete the key exchange protocol using a replay attack. To create message 6 the attacker needs the session key. The attacker cannot recreate the session because of assumption 3, the client generates the session key using a PUF. PUF is a physical entity, which resides within the client and its behavior can never be replicated by the attacker. Therefore, if the client receives message 6, then the sender is guaranteed to be the trusted server and the real sender of the message.

3.3.5. Guarantee 5

If the server receives the encrypted finished message, the client proves that it can use the session key it sent in the message 5. After sending the message 6, the server listens to an encrypted message from the client that reads as “finished” after decryption. An attacker trying to pose as the client has 2 options: replaying the or creating the message 7. Each session has a unique session key encrypted message 6. If the attacker replays the captured message 7, the server will not obtain the “finished” message after decryption for the same reasons explained in proof 3. Therefore, the Attacker cannot complete the key exchange protocol using a replay attack. To create the message 7 the attacker needs the session key. By assumption 3, the attacker can never recreate the session key. Thus, the attacker cannot create the encrypted “finished” message. If the server successfully gets this message, it guarantees that the client which sent the session key can also use it.

4. EXPERIMENTAL WORK

This section will present information and discussion of the IoT network simulation performed to measure the effectiveness of the proposed handshake protocol in detail. The network simulation includes a central CoAP server and 30 IoT nodes. By utilizing the device information obtained through the handshake protocol, the server rejects certain devices and only allows devices which satisfy certain specifications to enter to the network. By performing autonomous real-time anomaly detection using machine-learning, the server aims to detect anomalous or malicious network traffic and block further network traffic generated by suspected malicious nodes. This section will provide the dataset used to train the classifiers, while also comparing five different machine learning classifiers, which were considered based on their performance and suitability for real-time anomaly detection. This section will also discuss the simulation environment, results obtained from the simulation, as well as how the preferred classifier performed on the simulation.

4.1. The Dataset Description

The IoTID20 dataset is used as the main source of data [26]. The data is generated from a common smart home setup where victim and attacking devices are present. The IoTID20 data contains over 625,000 instances which consists of 80 network features and 3 label features which can be seen below as:

- Binary: Normal, Anomaly
- Category: Normal, DoS (Denial of Service), Mirai, MITM (Man in the Middle), Scan
- Subcategory: Normal, Syn Flooding, Brute Force, HTTP Flooding, UDP Flooding, ARP Spoofing, Scan Host Port, Scan OS

[26] states that the most important benefit of the IoTID20 data is that it replicates a modern approach of IoT device communication, and it is among the few publicly available IoT intrusion detection datasets. The features present in the data is ranked using the Shapira-Wilk algorithm to measure the regularity of the distribution of instances with respect to the feature. They argue that more than 70% of the features ranked above 0.50 and state that these high ranked features will improve the classification capability of detection algorithms and techniques. To generate the data to be used for the network simulation, 20,000 instances were randomly chosen from each of the five main categories. To use the data within the network simulation, additional features were added which are not used by the classifiers during the training phase, and the dataset is revised to be compatible with the CoAP protocol.

4.2. Machine Learning Techniques for Autonomous Anomaly Detection

Machine learning is used for autonomous real-time anomaly detection in this work. Usage of machine learning classifiers for intrusion and anomaly detection for network security is thoroughly researched in the past decade [27], [28]. The classifiers studied in this paper include:

- Random Forest (RF)
- Decision Tree (DT)
- Stacking
- K-Nearest Neighbors (KNN)
- Gaussian Naïve Bayes (GNB)

DT, KNN and GNB classifiers are preferred due to their suitability for intrusion and anomaly detection [28], [29]. RF and Stacking classifiers are also included in this paper because they perform relatively well on the IoTID20 data as presented in [26]. The stacking classifier is used in this paper as the ensemble classifier.

4.2. Performance Evaluation of the Classifiers

The performance of each classifier on the test data can be seen from Table 2 and Table 3. It can clearly be seen that from the five different classifiers trained, the random forest, decision tree and the stacking classifiers performs the best on the IoTID20 data. The performance of the KNN classifier is similar with the RF and DT classifiers. However, it is stated in [30] that the KNN is not applicable for critical real time systems where high amounts of training samples are present. Furthermore, since the classification of a new instance x requires the calculation of all the distances between x and the training data in the KNN algorithm, it comes with a significant computational cost. Finally, the GNB classifier performed the worst considering the performances of other classifiers, as can be observed from Figure 3, Table 2 and Table 3.

Considering only the best performing three algorithms, namely the RF, DT and stacking classifiers, for the network simulation purposes, the stacking algorithm was preferred in this paper. Although the stacking algorithm is an ensemble classifier and it requires more computational power and resources for training, the difference in performance between the stacking algorithm and other better performing classifiers cannot be neglected. Moreover, from Figure 3, it is observed that the stacking algorithm is by far the best classifier for detecting scan port OS attacks on the IoTID20 data, outperforming other classifiers drastically.

Table 2. Macro avg. performances of the classifiers trained on category as target label.

Classifier	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Random Forest (RF)	0.90	0.93	0.90	0.90
Decision Tree (DT)	0.90	0.93	0.91	0.91
Stacking	0.91	0.93	0.92	0.92
K-Nearest Neighbor (KNN)	0.87	0.91	0.87	0.86
Naïve Bayes (GNB)	0.56	0.66	0.56	0.56

Table 3. Macro avg. performances of the classifiers trained on subcategory as target label.

Classifier	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Random Forest (RF)	0.82	0.73	0.74	0.71
Decision Tree (DT)	0.85	0.78	0.77	0.74
Stacking	0.89	0.86	0.85	0.84
K-N Neighbor (KNN)	0.78	0.71	0.72	0.69
Naïve Bayes (GNB)	0.48	0.46	0.43	0.40

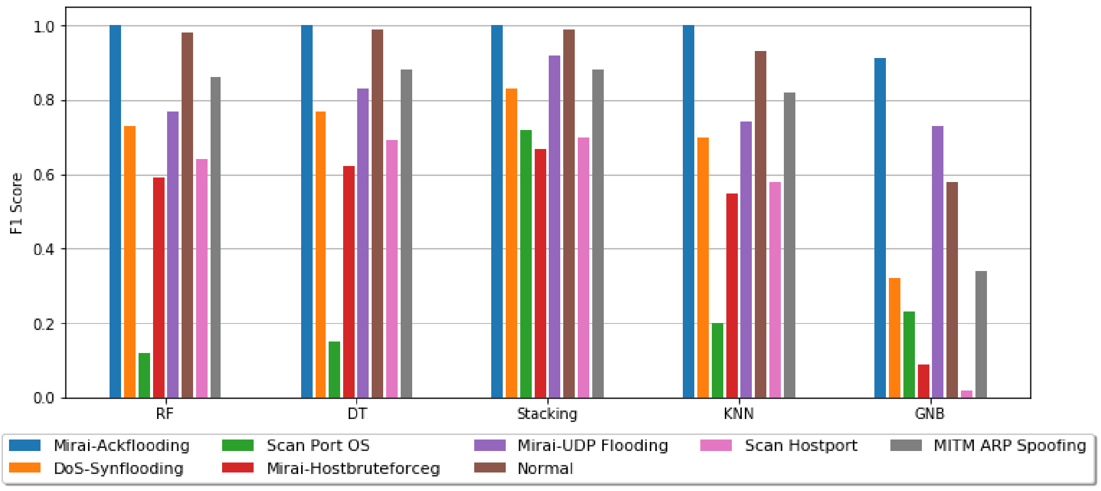

Figure 3. F1 Scores of the classifiers on the test data for subcategory label.

The classification duration is also considered when choosing the classifier to use during the network simulation. The stacking algorithm takes 0.06 ms to classify a single instance, whereas the decision tree classifier takes only 0.01 ms. This is to be expected due to the stacking classifier being computationally more expensive, but the difference between the two is quite negligible for the network simulation performed in this paper. However, if the number of IoT devices were to increase considerably in the network, the CoAP server could experience problems detecting anomalous network traffic using the stacking algorithm in real time. In such a case decision tree classifier should be preferred to stacking classifier.

4.3. Network Simulation

The network simulation which uses the proposed handshake protocol presented in this paper is implemented on Python 3.7. To simulate the distributed network a single central CoAP server, another server to act as the CA, and 30 IoT devices as the client nodes were used. The IoT devices range from simple and lower complexity devices such as smart cameras, controller and hubs, smoke alarms and printers, to more complex devices such as laptops, PCs, phones, and tablets. Each of these devices are given a unique client ID as required within the handshake protocol. The simulation assumes that none of these devices are previously known by the CoAP server, and all of them are required to perform the handshake protocol with the server to join the IoT network securely.

A total of 30 IoT devices are used to simulate the IoT network. Information and the assigned roles about these devices (see Table 4 and Table 5).

Table 4. Low complexity IoT devices used within the network simulation.

Device Name	Type	OS	ID	Role
Amazon Echo	Controller/Hubs	Nan	1	Victim
Belkin Motion Sensor	Energy Management	Nan	2	Victim
LIFX Light Bulb	Energy Management	Nan	3	Victim
iHome Power Plug	Energy Management	Nan	4	Victim
Belkin Switch	Energy Management	Nan	5	Victim
TP Link Power Plug	Energy Management	Nan	6	Victim
Netatmo Camera	Cameras	Nan	7	Victim
Nest Drop Camera	Cameras	Nan	8	Victim
Samsung Smart Camera	Cameras	Nan	9	Victim
TP Link Camera	Cameras	Nan	10	Victim
HP Envy Printer	Appliances	Nan	11	Victim
Pixstar Photo Frame	Appliances	Nan	12	Victim
Triby Speaker	Appliances	Nan	13	Victim
Withthings Sleep Sensor	Health-Monitor	Nan	14	Victim
Blipcare BP Meter	Health-Monitor	Nan	15	Victim
Netatmo Weather Station	Health-Monitor	Nan	16	Victim
Nest Smoke Alarm	Health-Monitor	Nan	17	Victim
Withthings Scale	Health-Monitor	Nan	18	Victim

Table 5. High complexity IoT devices used within the network simulation.

Device Name	Type	OS	ID	Role
Samsung Galaxy Tablet	Tablet	Android	19	Victim
Android Phone	Phone	Android	20	Victim
Laptop	PC	Ubuntu	21	Victim
MacBook	PC	Mac OS	22	Victim
Android Phone 2	Phone	Android	23	Victim
iPhone	Phone	iOS	24	Victim
MacBook 2	PC	Mac OS	25	Victim
iPhone 2	Phone	iOS	26	Malicious
iPhone 3	Phone	iOS	27	Malicious
MacBook 3	PC	Mac OS	28	Malicious
Laptop 2	PC	Windows	29	Malicious
Laptop 3	PC	Windows	30	Malicious

After generating a secure session with the CoAP server using the proposed handshake protocol, each device starts generating a network traffic. The devices were split into malicious/anomalous and benign devices as stated before. Each packet sent by the IoT devices were chosen from within the test data as discussed within the dataset section. Each device picks a random packet information to send from within its assigned instances in the test data. For ease of implementation and simulation, the information about the packets picked and sent by the devices are known by the CoAP server. This information is used by the server to perform real time anomaly detection using the trained stacking classifier as discussed before.

The CoAP server adopts a whitelisting approach. In the presented setup, higher complexity devices such as PCs, phones, laptops, and tablets are the whitelisted devices. As such, after the CoAP server receives the type and OS information of each device during the handshake protocol. The connection of devices which does not satisfy the whitelist policy described before are declined. Thus, the devices which satisfy the above conditions can generate a secure session with the CoAP server and are registered to the IoT network. Currently the type of a device and its respective OS is enough to be added into the IoT network. However, more specific whitelist policies can easily be added to tend to different scenarios if needed.

$$f(x) = \frac{\lambda^x}{x!} e^{-\lambda}$$

The network traffic each individual IoT device generates within the simulation follows a Poisson distribution. The Poisson distribution expresses the probability of a given number of events occurring in a fixed amount of time. It assumes that these events occur independently with a constant average rate. It is widely used for network and communication traffic simulations [31]. Poisson distribution can be expressed using the equation above where x denotes the number of occurrences and λ denotes the expected number of occurrences. A Poisson distribution with λ=3 packets sent per second by each client is used for the network simulation in this paper. It is important to mention that the λ can be any value as long as the CoAP server can handle the generated network traffic.

It is to be noted that the packet collision, loss, and faulty packets are omitted during the network simulation for convenience. The number of packets received by the CoAP server at each second throughout a 3-minute simulation can be seen from Fig. 4. The initial increase in the number of packets seen at the start of the simulation is due to each device performing the handshake protocol before being accepted to the IoT network. Due to the whitelisting policy, the connection of devices with IDs 1-18 are declined by the server during the handshake protocol. Thus, only the other 12 devices can join the IoT network and begin generating the network traffic. The number of packets at each second converges to around 36 as expected from 12 devices generating a network traffic based on a Poisson distribution with λ=3 packets sent per second.

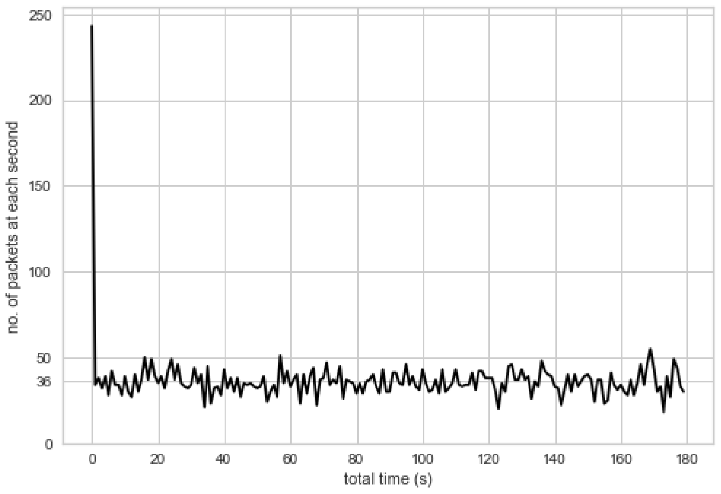

Figure 4. Number of packets received by the CoAP server at each second.

The CoAP server feeds the information about each packet within the IoT network traffic into the classifier for autonomous anomaly detection in real time. The binary performance of the classifier, which can be seen from Table 6, is measured as the classifiers ability to detect whether a packet is an anomaly or not. From Table 6, it is shown that the classifier performs almost perfectly when detecting anomalous packets, with a false negative rate of only 0.038%. The classifier classified some packets in the benign network traffic as anomalous, which can be seen from the misclassified instances detected from victim devices, yielding a false positive rate of 1.702%. The overall accuracy of the classifier in the binary setting is 99.98%, misclassifying only

67 packets out of 6591 packets in the IoT network simulation. The CoAP server inside the proposed network simulation setup was able to blacklist all the malicious devices with perfect accuracy whereas none of the benign devices were blacklisted by accident.

Table 6. Performance of the classifiers on detecting anomalous packets correctly from each device.

Device ID	Total Packets	Misclassified Packets	Accuracy
19	562	10	98.22%
20	558	15	97.31%
21	573	10	98.25%
22	556	9	98.38%
23	513	7	98.63%
24	557	8	98.56%
25	552	7	98.73%
26	553	0	100.00%
27	537	0	100.00%
28	549	0	100.00%
29	540	1	99.81%
30	541	0	100.00%

Table 7. Performance of the classifiers on correctly classifying packets from each device.

Device ID	Total Packets	Misclassified Packets	Accuracy
19	562	10	98.22%
20	558	15	97.31%
21	573	10	98.25%
22	556	9	98.38%
23	513	7	98.63%
24	557	8	98.56%
25	552	7	98.73%
26	553	47	91.50%
27	537	42	92.17%
28	549	59	89.25%
29	540	51	90.55%
30	541	55	90.57%

The classifiers performance on correctly classifying each packet can be seen from Table 7. Comparing both Table 6 and Table 7, since there is no additional classification of normal packets, the number of misclassified instances for victim devices remain the same. However, there are significantly more misclassified instances of the malicious devices. However, this does not mean that the classifier fails to detect anomalous packets, but rather fails to classify the type of the anomalies.

Considering both Table 6 and Table 7, the ability to detect the anomalous packets in the network traffic is enough for the CoAP server to maintain the security of the IoT network, and the classification of anomalies is not a high priority. Therefore, it can be said that the stacking classifier performs quite remarkably with an overall accuracy of 99.98%. Since the accuracy of the classifier is very high, a blacklisting approach can be used by the CoAP server to prevent malicious devices from generating network traffic within the IoT network.

Current limitations of the proposed system model include the assumption of secure verification between the IoT devices and the CA for server authentication. The model assumes a secure communication between the two which may be achieved through other handshake protocols like

TLS or a long term pre-shared secret which is inserted into the IoT devices and the CA. Both assumptions have their downsides in the proposed model. TLS uses traditional session key generation and public key encryption, which requires high computational power and hence not suitable for all IoT devices. Similarly, using a long term pre-shared secret is not suitable for the proposed use case and system model. Since any device capable of completing the proposed handshake protocol may enter the IoT network, distributing the pre-shared secret to the new devices poses a challenge.

The deficiencies of the proposed network simulation setup include the behavior of the malicious IoT nodes. The attacker IoT nodes within the network simulation generate high amounts of malicious network traffic which makes it relatively easy for the server to detect the malicious nodes since the classifier used for autonomous anomaly detection has a high performance as presented before. This may not be the case in a real-life scenario where the malicious devices actively try to impersonate benign devices by generating small amounts of malicious network traffic to avoid detection. In such a case, the proposed network simulation setup may be improved to include a more comprehensive trust-score based approach adopted by the CoAP server.

5. CONCLUSIONS

In this work, a secure, lightweight handshake protocol, designed to work on top of CoAP for the autonomous device discovery and management of IoT devices of any complexity is proposed. The protocol makes use of a PUF for session key generation, as well as whitelisting and blacklisting methods for ensuring the network security. To the best of our knowledge, the presented PUF-based handshake protocol is the first to perform blacklisting and whitelisting. Anomaly detection using machine-learning techniques is utilized for blacklisting. An in-depth security analysis of the proposed handshake protocol is provided, where the resilience of the protocol against forgery, replay and MITM attacks is shown with simulation results. The paper also provides and discusses on a network traffic simulation carried out using the proposed handshake protocol. By gathering the specifications of IoT nodes, the CoAP server can decide to only allow nodes join or remain in the IoT network that satisfy certain specifications using whitelisting and blacklisting methods. Additionally, a comparison of various machine learning algorithms performances and their fitness for real-time anomaly detection is provided. Amongst the five machine learning algorithms studied, the stacking algorithm is shown to display the highest level of accuracy of 99.98% in detecting anomalous data in the network. Based on the results of the network simulation performed, the CoAP server inside the proposed setup was able to blacklist all the malicious devices within the IoT clients with perfect accuracy.

Future work includes a comprehensive trust-score based blacklisting approach to further improve the performance of the proposed protocol. This improvement can make it easier for the server to detect malicious devices that impersonate benign devices by generating smaller amounts of malicious network traffic.

REFERENCES

[1] A. Rayes and S. Salam, Internet of Things From Hype to Reality, Springer International Publishing, 2019.

[2] A. Aktypi, K. Kalkan and K. B. Rasmussen, "SeCaS: Secure Capability Sharing Framework for IoT Devices in a Structured P2P Network," in Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy, New Orleans, Association for Computing Machinery, 2020, p. 271–282.

[3] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan and A.-R. Sadeghi, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 756-767.
[4] G. Fortino, F. Messina, D. Rosaci and G. M. Sarne, "ResIoT: An IoT social framework resilient to malicious activities," IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 5, pp. 1263-1278, 2020.
[5] G. Fortino, L. Fotia, F. Messina, D. Rosaci and G. M. Sarné, "Trust and Reputation in the Internet of Things: State-of-the-Art and Research Challenges," IEEE Access, vol. 8, pp. 60117-60125, 2020.
[6] G. Fortino and Mozilla, "The Transport Layer Security (TLS) Protocol Version 1.3.," 2018.
[7] Y. Yildiran, G. S. R and H. Basel, "Lightweight PUF-Based Authentication Protocol for IoT Devices," in 2018 IEEE 3rd International Verification and Security Workshop (IVSW), 2018, pp. 38-43.
[8] A. Bhattacharyya, T. Bose, S. Bandyopadhyay, A. Ukil and A. Pal, "LESS: Lightweight Establishment of Secure Session: A Cross-Layer Approach Using CoAP and DTLS-PSK Channel Encryption," in 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, 2015, pp. 682-687.
[9] A. Bin Rabiah, K. Ramakrishnan, E. Liri and K. Kar, "A Lightweight Authentication and Key Exchange Protocol for IoT," 2018.
[10] J. W. Byun, "End-to-End Authenticated Key Exchange Based on Different Physical Unclonable Functions," IEEE Access, vol. 7, pp. 102951-102965, 2019.
[11] B. Genge and J. W. Byun, "A Generic Multifactor Authenticated Key Exchange with Physical Unclonable Function," 2019.
[12] M. N. Aman, K. C. Chua and B. Sikdar, "Mutual Authentication in IoT Systems Using Physical Unclonable Functions," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1327-1340, 2017.
[13] A. Braeken, "PUF Based Authentication Protocol for IoT," Symmetry, vol. 10, no. 8, 2018.
[14] U. ChatterJee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata and M. P. Mukesh, "PUF + IBE: Blending Physically Unclonable Functions with Identity Based Encryption for Authentication and Key Exchange in IoTs.," 2017.
[15] N. Li, D. Liu and S. Nepal, "Lightweight Mutual Authentication for IoT and Its Applications," IEEE Transactions on Sustainable Computing, vol. 2, no. 4, pp. 359-370, 2017.
[16] A. Mostafa, S. J. Lee and Y. K. Peker, "Physical Unclonable Function and Hashing Are All You Need to Mutually Authenticate IoT Devices," Sensors, vol. 20, no. 16, 2020.
[17] Y. Ma, L. Yan, X. Huang, M. Ma and D. Li, "DTLShps: SDN-Based DTLS Handshake Protocol Simplification for IoT," IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3349-3362, 2020.
[18] J. Park and N. Kang, "Lightweight secure communication for CoAP-enabled Internet of Things using delegated DTLS handshake," in 2014 International Conference on Information and Communication Technology Convergence (ICTC), 2014, pp. 28-33.
[19] B. Anderson and D. McGrew, "Accurate TLS Fingerprinting using Destination Context and Knowledge Bases," 2020.
[20] S. F. Danilo, A. O. Hyggo and P. Angelo, "A personal connected health system for the Internet of Things based on the Constrained Application Protocol," Computers & Electrical Engineering, vol. 44, pp. 122-136, 2015.
[21] T. A. Alghamdi, A. Lasebae and M. Aiash, "Security analysis of the constrained application protocol in the Internet of Things," in Second International Conference on Future Generation Communication Technologies (FGCT 2013), 2013, pp. 163-168.
[22] D. Rathod, "Security Analysis of Constrained Application Protocol (CoAP): IoT Protocol," vol. 6, p. 37, 2017.
[23] Z. Shelby, K. A. Hartke and C. Bormann, "The Constrained Application Protocol (CoAP)," Universitaet Bremen TZI, 2014.
[24] A. Shamsoshoara, A. Korenda, F. Afgah and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for Internet of Things," Computer Networks, vol. 183, p. 107593, 2020.
[25] D. Dolev and A. Yao, "On the Security of Public Key Protocols," IEEE Transactions on Information Theory, 1983.
[26] I. Ullah and Q. H. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," in Advances in Artificial Intelligence, Cham, Advances in Artificial Intelligence, 2020, pp. 508-520.

[27] T. Mehmood, R. Md and B. Helmi, "Machine learning algorithms in context of intrusion detection," in 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), 2016, pp. 369-373.

[28] P. Illavarason and B. Kamachi Illavarason, "A Study of Intrusion Detection System using Machine Learning Classification Algorithm based on different feature selection approach," in 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, pp. 295-299.

[29] A. Srivastava, A. Agarwal and G. Kaur, "2019 4th International Conference on Information Systems and Computer Networks (ISCON)," in Novel Machine Learning Technique for Intrusion Detection in Recent Network-based Attacks, 2019, pp. 524-528.

[30] A. Starzacher and B. Rinner, "Evaluating KNN, LDA and QDA classification for embedded online feature fusion," 2009.

[31] B. Chandrasekaran, "Survey of Network Traffic Models," Washington University in St. Louis - Computer Science & Engineering at WashU, 1006.